CLAIM AMENDMENTS

Claim Amendment Summary

Claims pending

- Before this Amendment: Claims 1-9 and 11-54.

- After this Amendment: Claims 1-9, 11-40, and 42-54.

Non-Elected, Canceled, or Withdrawn claims: claim 41.

Amended claims: 1, 2, 4, 8, 13, 18, 25, 31-36, 40, 46, 48, 50, and 51.

New claims: None.

Claims:

1.   **(Currently amended)**   A method of generating common intermediate language code <u>for use in a framework, the method</u> comprising:

receiving a portion of JAVA[TM] language source code referencing [[a]]<u>, through a generic class syntax, one or more generic classes unspecified in a formal JAVA[TM] language specification, wherein:</u>

   <u>each of the one or more generic classes refers to a</u> first class ~~having a definition that is~~ <u>configured to operate</u> uniformly <u>on values of different types associated with the first class and defined by</u> ~~applicable to~~ a plurality of <u>second classes</u> ~~associated with the first class:~~ [[,]]

   the <u>generic class syntax is not specified in the formal JAVA[TM] language specification and identifies</u> ~~identifying~~ one of the plurality of <u>second classes</u>

~~associated classes~~ by surrounding the one of the plurality of second classes with angular brackets following the first class; and

~~, a definition of the first class being provided in a network using a first programming language other than JAVA™ language; and~~

generating, through a first compiler different from a formal compiler complying with the formal JAVA™ language specification, language-neutral intermediate language code representing the portion of JAVA™ language source code referencing the one or more generic classes ~~based on the portion of JAVA™ language source code and the first class provided in the first programming language other than JAVA™ language, wherein the language-neutral intermediate language code is generated at least from JAVA™ language and the first programming language.~~

2.   **(Currently amended)**   A method as recited in claim 1 further comprising parsing the portion of the JAVA™ language source code into a parse tree representing the portion of the JAVA™ language source code before compiling the portion with the first class.

3.   **(Original)**   A method as recited in claim 2 further comprising nesting a constructed class of the first class in the parse tree.

4.   **(Currently amended)**   A method as recited in claim 1 further comprising:

generating a parse tree having a token referencing the first class and a token referencing the specified one of the plurality of <u>second classes</u> ~~associated classes~~; and

semantically analyzing the parse tree to determine validity of semantics of the first class.

5.   **(Original)**   A method as recited in claim 4 wherein the semantically analyzing comprises determining whether operations applied to the first class are valid.

6.   **(Original)**   A method as recited in claim 1 further comprising generating metadata descriptive of the first class.

7.   **(Previously Presented)**   A method as recited in claim 6 further comprising storing the metadata with the language-neutral intermediate language code, whereby the language-neutral intermediate language code is used by an application program.

8.   **(Currently amended)**   A method as recited in claim 1 further comprising creating a compiled project including the language-neutral intermediate language code and metadata descriptive of the first class and the ~~specified~~ one of the plurality of <u>second classes</u> ~~associated classes~~.

9.   (Original)   A method as recited in claim 1 further comprising executing the language-neutral intermediate language code with a runtime engine.

10.   (Cancelled).

11.   (Previously Presented)   A method as recited in claim 1 wherein the framework is a .NET™ Framework.

12.   (Previously Presented)   A method as recited in claim 11 wherein the developing comprises authoring the portion of JAVA™ language source code with a VISUAL J# .NET™ application of the .NET™ Framework.

13.   (Currently amended)   A method of compiling <u>in a framework, the method</u> comprising:

receiving a portion of JAVA™ language software having a declaration of an instance of a generic class <u>unspecified in a formal JAVA™ language specification, the declaration of the instance of the generic class being implemented through a generic class syntax</u>, wherein:

the <s>declared instance of the</s> generic class <u>refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;</u>

the generic class syntax is not specified in the formal JAVA^{TM} language specification and identifies one of the plurality of second classes by surrounding the one of the plurality of second classes with angular brackets following the first class; and

~~is provided using a programming language other than JAVA^{TM} language;~~

parsing the declaration into a token corresponding to the generic class; and

creating an intermediate language code block corresponding to the parsed declaration through a first compiler other than a traditional compiler complying with the formal JAVA^{TM} language specification, wherein:

the intermediate language code block containing the instance of the generic classes ~~the intermediate language code block is generated based on the portion of JAVA^{TM} language and the instance of the generic class developed in the programming language other than JAVA^{TM} language and~~ is made executable by a runtime engine.

14.   (Original)   A method as recited in claim 13 further comprising associating the declaration of the instance of the generic class with a defined generic class in a generic class library.

15.   (Original)   A method as recited in claim 14 further comprising tokenizing a parse tree with an identifier corresponding to the defined generic class, the parse tree comprising a hierarchical representation of the declaration.

16.   (Original)   A method as recited in claim 13 further comprising creating metadata describing the portion of the JAVA™ language software.

17.   (Original)   A method as recited in claim 14 further comprising validating an operation on the instance of the generic class based on the defined generic class.

18.   (Currently amended)   A computer-readable medium having stored thereon computer-executable instructions for performing a method of compiling <u>in a framework, the method</u> comprising:

receiving a portion of JAVA™ language software including an instruction that references a generic class of a specified type <u>through use of a generic class syntax,</u> wherein<u>:</u>

the <s>referenced</s> generic class <u>is unspecified in a formal JAVA™ language specification and refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;</u>

<u>the generic class syntax is not specified in the formal JAVA™ language specification and identifies one of the plurality of second classes by surrounding the one of the plurality of second classes with angular brackets following the first class; and</u>

<s>is provided using a programming language other than JAVA™ language;</s>

creating a parse tree having a generic class identifier associated with the generic

class and type identifier associated with the specified type; and

generating, through a first compiler other than a traditional compiler complying

with the formal JAVA™ language specification, one or more intermediate language

instructions representing the JAVA™ language instruction based on the parse tree ~~based~~

~~on the portion of JAVA™ language source code and the referenced generic class~~

~~provided in the programming language other than JAVA™ language~~.

19.   **(Original)**   A computer-readable medium as recited in claim 18, the

method further comprising translating the one or more intermediate language instructions

into microprocessor-specific binary for execution by a computer.

20.   **(Original)**   A computer-readable medium as recited in claim 18, the

method further comprising validating the parse tree according to a generic class definition

associated with the generic class.

21.   **(Original)**   A computer-readable medium as recited in claim 20, wherein

validating the parse tree comprises determining whether an assignment applied to the

instance of the generic class assigns an allowable type to the instance.

22.   **(Original)**   A computer-readable medium as recited in claim 18, the

method further comprising generating metadata associated with the generic class.

23. **(Original)** A computer- readable medium as recited in claim 18, wherein the specified type is a second generic class of a second specified type.

24. **(Original)** A computer- readable medium as recited in claim 23, wherein the method further comprises nesting the second generic class and the second specified type at different levels in a hierarchy in the parse tree.

25. **(Currently amended)** A computer-readable medium having stored thereon intermediate language code block executable by a runtime engine, the intermediate language code block generated from a representation of a portion of source code for use by a compiler in a framework, the representation comprising:

a generic class identifier field having data identifying, through use of a generic class syntax, a generic class referenced in the portion of source code in a first programming language for which [[a]] the generic class syntax is not formally specified, wherein:

the referenced generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes; and

the generic class syntax identifies one of the plurality of second classes by surrounding the one of the plurality of second classes with angular brackets following the first class; and

is provided using a second programming language other than the first programming language; and

a constructed class identifier field having data identifying a constructed class of the generic class.

26.   **(Previously Presented)**   A computer-readable medium as recited in claim 25, wherein the representation further comprises:

at least one nested constructed class that is a generic class.

27.   **(Original)**   A computer-readable medium as recited in claim 25, wherein the generic class identifier identifies a Queue class.

28.   **(Previously Presented)**   A computer-readable medium as recited in claim 25, wherein the first programming language is a JAVA^{TM} language.

29.   **(Previously Presented)**   A computer-readable medium as recited in claim 25, wherein the representation further comprises metadata describing the generic class.

30.   **(Original)**   A computer-readable medium as recited in claim 25, wherein the constructed class comprises one of:

an integer type;

a float type;

a Stack type;

a Queue type; and

a Dictionary type.

31. **(Currently amended)** A method [[for]] of compiling in a framework, the method comprising:

receiving a portion of source code in a first programming language for which one or more .NET™ generic types are not specified in a formal definition of the first programming language, wherein:

each of the one or more the specified generic types [[are]] refers to a first type configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second types:

each of the one or more generic types uses a generic type syntax not specified in the formal definition of the first programming language:

provided using a second programming language other than the first programming language;

parsing the portion of source code into a parse tree having comprising each [[an]] instance of the one or more generic types in the portion of source code, wherein each instance of the one or more generic types comprises:

[[a]] the first type; and having

at least one instance of <u>one of the plurality of second types associated with</u> <u>the first type; and</u>

~~an associated type; and~~

generating an intermediate representation of the parse tree representing the <u>parse</u> <u>tree</u> ~~source code in the first programming language and the instance of the associated type~~ ~~using the second programming language.~~

32. **(Currently amended)** The method as recited in claim 31 further comprising importing metadata describing the first [[class]] <u>type</u> and the at least one instance of <u>one of the plurality of second types associated with the first type</u> ~~the~~ ~~associated class.~~

33. **(Currently amended)** The method as recited in claim 31 further comprising tokenizing the parse tree with a token corresponding to the <u>one or more</u> <u>generic types</u> ~~generic type.~~

34. **(Currently amended)** The method as recited in claim 33 further comprising tokenizing the parse tree with at least one token corresponding to the at least on instance of <u>one of the plurality of second types associated with the first type</u> ~~the~~ ~~associated type.~~

35.   (Currently amended)   The method as recited in claim 31 wherein each of the one or more generic types generic type is a .NET™ generic class.

36.   (Currently amended)   A method of generating microprocessor-executable code in a framework, the method comprising:

receiving a portion of source code written in a first programming language for which generic classes are unspecified, the portion of source code including a generic class declaration declaring a generic class, the generic class declaration including at least one associated class reference defining a constructed class of the generic class, wherein:

the declared generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes:

the generic class uses a generic class syntax not specified in a formal specification of the first programming language:

the generic class declaration creates a constructed class of the generic class by associating a reference of one of the plurality of second classes with the generic class: and

is provided using a second programming language other than the first programming language; and

generating a module having microprocessor-executable instructions corresponding to the constructed class based on the portion of source code in the first programming language and the associated class reference of the generic class using the

second programming language, the module further having metadata describing the constructed class.

37.   (Original)   A method as recited in claim 36 wherein the microprocessor-executable instructions comprise intermediate language instructions.

38.   (Original)   A method as recited in claim 36 wherein the microprocessor-executable instructions comprise Microsoft® Intermediate Language instructions.

39.   (Original)   A method as recited in claim 36 wherein the metadata comprises at least one of:

a name of the constructed class;

visibility information indicating the visibility of the constructed class;

inheritance information indicating a class from which the constructed class derives;

interface information indicating one or more interfaces implemented by the constructed class;

method information indicating one or more methods implemented by the constructed class;

properties information indicating identifying at least one property exposed by the constructed class; and

events information indicating at least one event the constructed class provides.

40.   **(Currently amended)**   A method of compiling in a framework, the method comprising:

receiving a portion of source code written in a first programming language for which one or more generic classes are unspecified in a formal language specification of the first programming language, ~~the portion of source code including a first class reference having at least one associated class reference referencing a class associated with the first class,~~ wherein:

each of the one or more generic classes refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;

each of the one or more generic classes uses a generic class syntax unspecified in the formal language specification of the first programming language; and

~~the first class reference is provided using a second programming language other than the first programming language; and~~

generating an intermediate language representation of the portion of the source code in the first programming language through a first compiler other than a traditional compiler complying with the formal language specification of the first programming language, the intermediate representation having an instance of the first

class ~~provided in a second programming language~~ and an instance of the at least one <u>of</u>

<u>the plurality of second classes</u> ~~associated class~~.

41. **(Cancelled).**

42. **(Previously Presented)** A method as recited in claim 40 wherein the first programming language is a JAVA^{TM} language.

43. **(Original)** A method as recited in claim 40 further comprising validating the type based on a definition of the first class.

44. **(Original)** A method as recited in claim 43 further comprising validating an operation on the first class based on a definition of the first class.

45. **(Original)** A method as recited in claim 40 further comprising interpreting the intermediate representation for execution by a microprocessor.

46. **(Currently amended)** A method as recited in claim 40 wherein angular brackets surround the at least one <u>of the plurality of second classes</u> associated

<u>with a reference of the first</u> class ~~reference~~.

47.   (Original)   A method as recited in claim 40 wherein the first class is a Queue class.

48.   (Currently amended)   A method as recited in claim 47 wherein the at least one of the plurality of second classes ~~associated class~~ comprises ~~at least~~ one of:

an int type;

a string type; and

a Queue type.

49.   (Original)   A method as recited in claim 48 wherein the Queue type includes at least one nested class reference referencing a second type associated with the Queue type.

50.   (Currently amended)   A method as recited in claim 40 wherein the at least one of the plurality of second classes associated with a reference of the first class ~~reference~~ includes one or more nested generic class references.

51.   (Currently amended)   A system for compiling in a framework, the system comprising:

a parser receiving JAVA™ language source code having an instruction referencing a generic class in a generic class syntax ~~provided in a first programming language other than JAVA™ language~~ and specifying a type of the generic class, the

parser further creating a parse tree from the JAVA™ language source code, the parse tree including a first node representing the generic class and a second node representing the specified type of the generic class, wherein :

the generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;

the generic class syntax is unspecified in the formal language specification of JAVA™ programming language and supported in the framework; and

a code generator generating intermediate language code representing the JAVA™ language source code referencing the generic classes ~~based on the JAVA™ language source code and the referenced generic class provided in the first programming language~~.

52. **(Original)** A system as recited in claim 51 further comprising:

a common intermediate language importer providing tokens associated with the generic class and the specified type of the generic class.

53. **(Original)** A system as recited in claim 51 further comprising a runtime engine executing the intermediate language code.

54. **(Original)** A system as recited in claim 51 further comprising a semantic analyzer analyzing the specified type to determine whether the specified type is an allowable type of the generic class.

Serial No.: 10/657,463
Atty Docket No.: MS1-1596US
Atty/Agent: Ningning Xu
-19-
lee&hayes   The Business of IP™
www.leehayes.com   509 324.9256